SOARE BOGDAN 441A

DEZVOLTAREA RAPIDĂ A APLICAȚIILOR

Prezenatare Generală

Dezvoltarea rapidă a aplicațiilor (Rapid Application Development sau RAD) este o metodologie de dezvoltare software care utilizează planificarea minimă în favoarea prototipurilor rapide. Planificarea software-ului dezvoltat folosind RAD este intercalată cu scrierea software-ului în sine. Lipsa de o pre-planificare extinsă, în general permite ca software-ul să fie scris mult mai rapid, și este mai ușoară schimbarea cerințelor. [1]

Termenul a devenit recent un buzzword de marketing care descrie generic aplicații care pot fi concepute și dezvoltate în termen de 60-90 de zile [4], dar a fost inițial destinat pentru a descrie un proces de dezvoltare care presupune aplicarea prototipurilor și dezvoltarea iterativă.

În dezvoltarea rapidă de aplicații, tehnicile structurate și prototipurile sunt în special folosite pentru a defini cerințele utilizatorilor și pentru a proiecta sistemul final. Procesul de dezvoltare începe cu dezvoltarea de modele preliminare de date folosind tehnici structurate. În următoarea etapă, cerințele sunt verificate folosind prototipuri, pentru a rafina eventual datele și modelele de proces. Aceste etape sunt repetate iterative. [1]

Abordarea RAD poate implica compromisuri în funcționalitate și performanță în schimbul unei dezvoltări rapide și a usurinței întreținerii aplicației.

Istorie

RAD se află în existența de aproape 20 de ani, dar este la fel de valabil astăzi așa cum a fost atunci când a fost inițial conceptualizat.

Termenul Rapid Application Development (RAD) a fost folosit și implementat pentru prima dată cu succes în timpul anilor 1970 de către New York Telephone Co's Systems Development Center sub indrumarea lui Dan Gielan. [5]

RAD implică dezvoltarea iterativă și construcția de prototipuri. În 1990, în cartea sa RAD, Rapid Application Development, James Martin a documentat interpretarea sa a

metodologiei. Mai recent, termenul și acronimul său au ajuns să fie folosite într-un sens mai larg.

Procesele dezvoltate în anii 1970, cum ar fi metodologia Waterfall de dezvoltare, de multe ori a rezultat în dezvoltarea de aplicații care nu corespundeau cerințelor clienților, deoarece aplicațiile durau atât de mult timp pentru a fi construite încât cerințele se schimbau înainte ca sistemul să fie complet.

Cauza problemei a fost identificată în stricta respectare a terminării unei etape din ciclul de viață înainte de a trece la etapa următoare a ciclului de viață. Concret, construirea unei aplicații bazată pe niște cerințe care au fost înghețate la un anumit punct în timp înseamnă că dezvoltarea durează mai mult imp, probabil că nevoile de business-ului se vor schimba și vor invalida cerințele pe care sistemul dezvoltat se bazează.

Au existat multe răspunsuri la această problemă din 1980 până astăzi. În 1986 Barry Boehm a scris ”A Spiral Model of Software Development and Enhancement”, care definește inițial conceptele de prototipuri și dezvoltarea iterativă și a avut un accent pe reducerea riscurilor. În timpul anului 1980 Scott Shultz și James Martin au rafinat ideile de prototipuri și dezvoltarea iterativă într-o metodologie numită ”Rapid Iterative Production Prototyping” care s-a concentrate pe dezvoltarea sistemelor într-un interval de timp scurt, cu echipe mici formate din personal înalt calificat, motivat, și cu experiență. James Martin (care a fost nominalizat pentru un premiu Pulitzer) a extins și formalizat în continuare ”Rapid Iterative Production Prototyping” și în 1991 a publicat cartea ”Rapid Application Development”.[5]

Avantaje și dezavantaje

Viteza și calitatea sunt avantajele principale ale Dezvoltării rapide a aplicațiilor, în timp ce scalabilitate redusă și seturile de caracteristici sunt dezavantajele.

1. Creșterea vitezei

După cum sugerează și numele, avantajul principal RAD constă creșterea vitezei de dezvoltare și a timpului scăzut de livrare. Scopul de a oferi aplicații rapid se adresează prin utilizarea de Inginerie Software asistată de calculator sau instrumente CASE, care se concentrează pe conversia cerințelor în cod cât mai repede cât posibil. [5]

2. Creșterea calității

Creșterea calității este un obiectiv principal al metodologiei RAD, dar termenul are un alt sens decât ce este în mod tradițional asociată cu dezvoltarea de aplicații personalizate. Conform RAD, calitatea este definită atât după gradul în care o aplicație livrată îndeplinește nevoile utilizatorilo cât și ca gradul costurilor scăzute ale sistemului. [5]

RAD încearcă să livreze calitate prin implicarea de utilizatori în analiza și în special fazele de proiectare.

3. Scalabiliatate redusă

Deoarece RAD se concentrează pe dezvoltarea unui prototip, care este iterativ dezvoltat într-un sistem complet, soluției livrate îi poate lipsi scalabilitatea unei soluții care a fost concepută ca o aplicație completă de la început. [5]

La arhitectura automată metodologia Just-In-Time oferă beneficiile Dezvoltării rapide a aplicațiilor minimizând în același timp multe dintre dezavantaje, cum ar fi scalabilitate redusă.

4. Caracteristici reduse

În cazul în care caracteristicile sunt lasăte pentru versiunile ulterioare, în favoarea livrării unei cereri într-un timp scurt, RAD poate produce aplicații care sunt mai puțin complete decât aplicațiile dezvoltate în mod tradițional. Această problemă ar trebui să fie tratată cât mai curând posibil, printr-o comunicare clară cu clientul asupra ce aplicație va fi livrată și când. [5]

Elemente de bază ale RAD

Dezvoltarea rapidă a aplicațiilor nu este potrivită pentru toate proiectele. Metodologia funcționează cel mai bine pentru proiecte în cazul în care domeniul de aplicare este mic sau munca poate fi împărțită în bucăți mici ușor de gestionat. De-a lungul acestor linii, echipele de proiect trebuie să fie, de asemenea, mici - preferabil 2-6 persoane - și trebuie să aibă experiență în toate tehnologiile care urmează să fie utilizate.

Obiectivele business-ului vor trebui să fie bine definite înainte de a putea începe proiectul, astfel încât proiectele care utilizează RAD nu ar trebui să aibă un domeniu de aplicare larg sau slab definit. În plus, în scopul de a menține proiectul într-un interval de timp scurt, deciziile trebuie să fie în măsură să se facă rapid, ceea ce înseamnă ca există foarte puțini factori de decizie ai clientului (de preferat numai unul), iar acestea trebuie să fie identificate în mod clar în față.

1. Prototipul

Un aspect cheie al RAD este construirea unui prototip, obiectivul este de a construi o versiune cu funcționalități reduse a produsului finit în cel mai scurt timp posibil, de preferință zile. Prototipul inițial servește ca o dovadă a conceptului de client, dar mai important servește ca un punct de vorbit și un instrument pentru rafinarea cerințelor.

Dezvoltarea de prototipuri rapid se realizează cu ajutorul instrumentelor ”Computer Aided Software Engineering CASE” care se concentrează pe captarea cerințelor, convertind modelul de date la o bază de date, și generarea codului, toate într-un singur instrument. Instrumente CASE au fost populare la sfârșitul anilor 80 și începutul anilor 90, dar cum tehnologia s-a schimbat câteva instrumente profită din plin de întregul potențial al tehnologiei instrument CASE. [5]

2. Dezvoltarea iterativă

Dezvoltarea iterativă înseamnă crearea de versiuni din ce în ce mai funcționale ale unui sistem în cicluri scurte de dezvoltare. Fiecare versiune este revizuită cu clientul pentru a produce cerințele pentru următoarea versiune. Procesul se repetă până când întreaga funcționalitate a fost dezvoltată.Lungimea ideală de iterații este între o zi și trei săptămâni.

Fiecare ciclu de dezvoltare oferă utilizatorului posibilitatea de a oferi feedback, rafina cerințele, și pentru a vizualiza progresul. În cele din urmă dezvoltaread iterative este cea care rezolvă problemele inerente metodologii inflexibile create în 1970. [5]

3. Time boxing

”Time boxing” este procesul de selectare a caracteristicilor care urmează să fie lansate în iterațiile ulterioare, în scopul de a finaliza versiunea curentă într-un interval

scurt de timp. Importanța acestui pas este de a nu fi subestimat. Fără un time boxing strict durata ciclurilor de repetare va crește, minimizând beneficiile RAD. [2]

4. Membrii de echipă

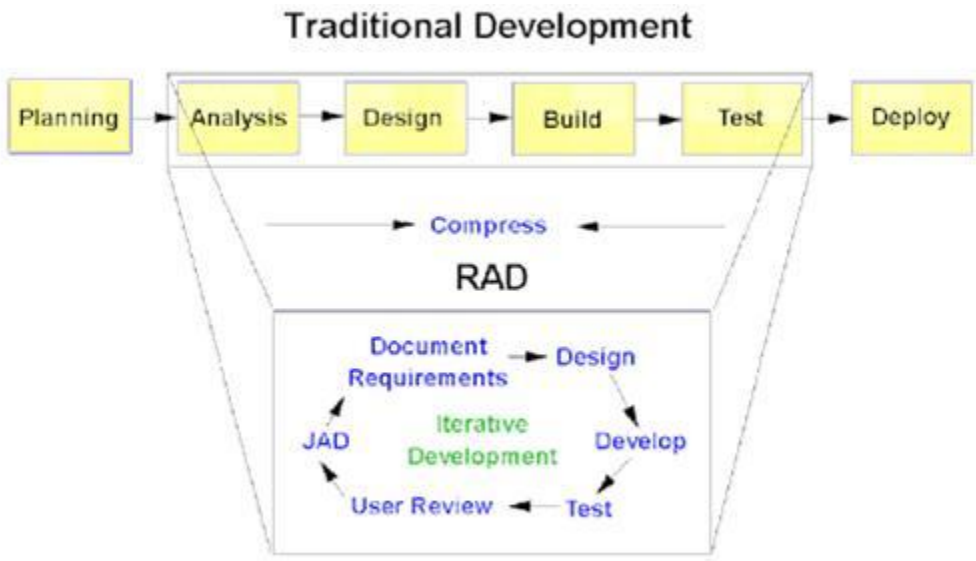
Metodologia RAD recomandă utilizarea unor echipe mici, care constau din membrii versatili cu experiență, și motivați care sunt în măsură să îndeplinească roluri multiple. Clientul joacă un rol vital în procesul de dezvoltare, resursele dedicate clientului trebuie să fie disponibile în timpul sesiunii de dezvoltări inițiale, precum și sesiuni de focus grup care au loc la finalul ciclului de dezvolatare. [5]

5. Managementul de proiect

Managementul de proiect activ și implicat este vital pentru a atenua riscurile de cicluri de dezvoltare lungi, neînțelegeri cu clientul, și termene ratate. Datorită iteratii scurte, managementul trebuie să fie axat pe doborârea tuturor obstacolelor care există la sfârșitul business-ului și garantând implicarea activă a clientului în realizarea de prototipuri, scrierea de cazuri de testare. [5]

6. Tehnologie

Când RAD a fost prima dată conceput era considerat important să se utilizeze instrumente care încorporează cele mai recente tehnologii pentru a accelera dezvoltarea. Acest lucru nu este mai puțin adevărat astăzi, alegerea tehnologiei pentru a pune în aplicare un proiect este un factor important în maximizarea beneficiilor RAD. Cum industria combină cele mai bune practici din diferite strategii, noi tehnologii evoluează care permit realizarea de prototipuri mai rapid,o mai bună modelare.



[6]

Metodologii RAD

De-a lungul anilor, mai multe forme diferite de RAD au apărut, fiecare cu propriile lor tehnici și concentrări. Următoarea este o listă scurtă de metodologii populare și a obiectivelor lor principale: [3]

Agile software development (Agile)	
Avantaje	Minimizarea "feature creep" prin intervale scurte de timp și mini-incrementare de lansări
Dezavatanje	Iterațiile scurte pote adăuga funcționalitate prea puțină, ceea ce duce la întârzieri semnificative în iterațiile finale. Deoarece Agile subliniază comunicare în timp real, folosind-o este problematic pentru o echipă mare de dezvoltare. Metodele Agile produc foarte puțină documentație scrisă și necesită o cantitate semnificativă de documentație post-proiect.
Extreme Programming (XP)	
Avantaje	Reduce costurile de modificări, prin spirale rapide ale noilor cerințe. Activitatea de proiectare apare progresiv.
Dezavatanje	Programatorii trebuie să lucreze în perechi, ceea ce este dificil pentru unii oameni.
Joint application design (JAD)	
Avantaje	Capturează vocea clientului prin implicarea acestuia în proiectarea și dezvoltarea aplicației printr-o serie de workshop-uri numite sesiuni JAD.

Dezavatanje	Clientul poate crea o viziune nerealistă a produsului și solicita lucruri în plus , rezultând că echipa va supra sau sub-dezvolta funcționalitatea aplicației.
Lean software development (LD)	
Avantaje	Creează soluții minimaliste și furnizează funcționalitate mai puțină, pe politica ca 80% azi este mai bine decât 100% mâine.
Dezavatanje	Produsul își poate pierde avantajul competitiv din cauza faptului că funcționalitatea de bază este insuficientă și din cauza slabei calități globale.
Rapid application development (RAD)	
Avantaje	Promovează o atmosferă puternică de colaborare și colectarea dinamică a cerințelor. Proprietarul de afaceri participă activ în realizarea de prototipuri, scrierea cazurilor de testare.
Dezavatanje	Dependența de echipe bine antrenate, puternice și angajamentul individual în proiect.
Scrum	
Avantaje	Îmbunătățirea productivității în echipe anterior paralizată "procesul" greu, abilitatea de a prioritiza munca.
Dezavatanje	Bazarea pe un șef care e posibil să nu dispună de abilitățile politice pentru a elimina obstacolele și să livreze scopul aplicației. Bazarea pe auto-organizarea echipelor poate conduce la lupte interne pentru putere și poate paraliza o echipă.

Procesul

1.Planificarea cerințelor

Combină elemente ale sistemului de planificare și faza de analiză a sistemelor de ciclul de viață Sistemul de Dezvoltare (SDLC). Utilizatorii, manageri și membri ai personalului IT discută și să cad de acord asupra nevoilor de afaceri, domeniul de aplicare proiectul, constrângeri, și cerințele de sistem. Se termină atunci când echipa este de acord cu privire la aspectele cheie și obține autorizația de management pentru a continua.

2. Proiectarea de utilizare

În timpul acestei faze, utilizatorii interacționează cu sistemele de analisti și de a dezvoltă modele și prototipuri care reprezintă toate procesele de sistem, intrările, ieșirile.

Grupurile de RAD sau subgrupe de obicei folosesc o combinație comună de dezvoltare de aplicatii (JAD) tehnici și instrumente CASE pentru a traduce nevoile utilizatorilor în modele de lucru. Proiectare utilizator este un proces continuu interactiv ce permite utilizatorilor să înțeleagă, să modifice, în cele din urmă și să aprobe un model de lucru a sistemului, care să răspundă nevoilor lor.

3. Constructie

Se concentrează pe dezvoltarea de aplicații și sarcini similare cu SDLC. În RAD, cu toate acestea, utilizatorii continuă să participe și pot sugera în continuare modificări sau îmbunătățiri ca ecranele reale sau rapoartele sunt dezvoltate. Sarcinile sale sunt de programare și dezvoltare de aplicații, codificare, unitate de-integrare și testare a sistemului.

Este vital să se păstreze fiecare iterație de dezvoltare pe drumul cel bun, și funcționalitate poate fi redusă pentru a menține dezvoltarea în time box. Managementul joacă un rol vital în asigurarea faptului că totul progresează conform planului, menținerea la current a clientului referitor la modificările funcționalității, și menținerea echipei motivată.

După ce prototipul a fost dezvoltat, echipa de construcție testează prototipuri inițiale cu ajutorul scripturilor de testare dezvoltate în timpul etapei de proiectare de utilizare, echipa de design face o recenzie asupra aplicației, clientul face și el o recenzie aplicației și în final echipa de construcție, echipa de design și clientul se întâlnesc în grupuri focus, în scopul de a stabili cerințele pentru următoarea iterație.

4. Implementare

Etapa de implementare cuprinde integrarea noului sistem în proiectul de afaceri. Echipa de dezvoltare pregătește de datele (cum ar fi valori de căutare, cum ar fi state și țări) și implementează interfața altor sisteme. Echipa de design antrenează utilizatorii de sistem, în timp ce utilizatorii efectuează testele de acceptare. Echipa de proiectare îi ajută pe utilizatori să transfer de proceduri lor vechi la cele noi care implică noul sistem, descoperă probleme după implementare, identifică și urmărește potențialele îmbunătățiri. Timpul necesar pentru a finaliza faza de implementare variază în funcție de proiect.

5. Activități post-proiect

Ca și cu orice proiect final livrabil, toate informațiile și datele despre proiect ar trebui să fie predate clientului. Concret, este o practică bună pentru un manager de proiect revizuirea și documentarea proiectului, organiza și stocare părților din proiect, cum ar fi componente de cod reutilizabile, planul de proiect, precum și planul de testare.

Popularitate

Industria de software a cunoscut o schimbare în strategia de dezvoltare, trecerea de la bazată pe sesiunea client / server de dezvoltare la o dezvoltarea în colaborare fără sesiune. Acest lucru, împreună cu utilizarea tot mai mare de framework-uri open source și produse din dezvoltarea comercială de bază, a reaprins interesul pentru metodologiile RAD pentru mulți dezvoltatori. [2]

Toate formele de RAD au potențial pentru a furniza un schelet propice pentru dezvoltarea de produse rapid, cu o calitate îmbunătățită a codului, dar punerea în aplicare cu succes și beneficiile depind de multe ori de tipul de proiect, program, ciclul software, și cultura corporativă.

În ultimii ani Agile / Scrum și Extreme Programming au primit multă atenție. Un lucru interesant comun între cele două abordări este faptul că calitatea codului joacă un rol-cheie în atât mentenabilitate și aplicarea în continuare a RAD, fără pierderi de viteza de dezvoltare în iterații ulterioare.

Concluzii

Metodele tradiționale de dezvoltare sunt adesea criticate de către dezvoltatorii de software pentru inflexibilitatea cauzată de dependența de o singură analiză de cerințe. Dezvoltarea rapidă a aplicațiilor propune o abordare care vizează servirea mai bună a nevoilor pieței software-ului permițând schimbarea folosind prototipuri, iterarea, și concepte Timeboxing. [2]

În ultimii ani accentul a RAD a evoluat mai mult către metode practice de prototipuri, iterații mai rapide, prin generatoare de cod mai bune, precum și utilizarea unor framework-uri de aplicații.

În ciuda popularității sale, există foarte puține pachete software care pot sprijini pe deplin tehnicile RAD. Multe instrumente de modelare nu pot genera cod, generatoarele de cod folosesc modele nepractice care nu pot genera o aplicație completă .

Dezvoltarea rapidă a aplicațiilor în mod clar este mai mult decât livrarea aplicațiilor cât mai repede posibil. James Martin a conceput-o ca o abordare bine definită pentru a dezvoltarea de aplicații care implică cicluri de dezvoltare scurte și iterative, time-boxing, prototipuri, precum și utilizarea de tehnologii moderne pentru captarea cerințelor. [4]

Această abordare este la fel de valabilă și astăzi cum a fost la sfârșitul anilor 1980, doar sensul său a fost oarecum ascuns prin introducerea pe piață modernă.

Bibliografie

[1] - https://en.wikipedia.org/wiki/Rapid_application_development

[2] - http://www.novulo.com/Rad.aspx

[3]https://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Rapid_Application_Development

[4] - What is Rapid Application Development, CASEMaker Totem

[5] - Rapid Application Development, Republished from Developers.net

[6] - http://elanemergingtechnologies.blogspot.ro/2010/08/rapid-application-development.html